Operating Systems Assignment - Technical Documentation

Table of Contents

Introduction

This project was created as part of the Operating Systems course at IMC FH Krems. Goal of this programming project is to implement analternative Linux shell in C with the following features:

- exec
- global usage
- quit
- background processes
- redirection of output

Installation

To install this project, you need to clone the repository and compile the source code. The following commands will clone the repository and compile the source code:

```
git clone

cd os_project

make
```

Usage

To use this project, you need to run the executable file. The following command will run the executable file:

```
./mine
```

Examples

The following examples show how to use the features of this project.

exec ps - uax

This command will execute the command ps - uax and redirect the output to the console

exec ps - uax > output.txt

This command will execute the command ps - uax and redirect the output to the file output.

exec la > out.LOG &

This command will execute the command ls -a in the background and redirect the output to the file called out.LOG.

global usage

This command will show a string: IMCSH Version 1.1 created by Bertold Vinze, David Bobek and Dinu Scripnic

quit

This command will quit the shell. If there are any running background processes, it will ask the user if he wants to terminate them and then quit the shell.

Technical details

This project was created using the following technologies:

- C
- Makefile

Process Struct:

- This struct is used to store the information about the processes that are running in the background

```
typedef struct process {
    pid_t pid;
    char *name;
} process;
```

Main function:

- Initializes the shell
- Infinite loop gets the user input
- Tokenizes the input
- Passes the tokens to the commandHandler

```
char line[MAXLINE];
char *tokens[LIMIT];
...
```

```
// create processes array
process *processes[100];
...
static char *currentDirectory;
setenv("shell", getcwd(currentDirectory, 1024), 1);
...
while (TRUE)
{
    fputs("IMCSH> ", stdout);
    ...
    memset(line, '\0', MAXLINE);
    ...
    // We wait for user input
    fgets(line, MAXLINE, stdin);
    ...
    // We read all the tokens of the input and pass it to our
    // commandHandler as the argument
    numTokens = 1;
    while ((tokens[numTokens] = strtok(NULL, " \n\t")) != NULL)
        numTokens++;
    commandHandler(tokens,processes);
}
```

Command Handler:

- Maps the command to the corresponding function

```
void commandHandler(char *tokens[],process *processes[])
{
    ...
    else if (strcmp(tokens[0], "exec") == 0)
    {
        executeFunction(tokens,processes);
    }
    ...
}
```

Execute Function:

- Forks the process
- Checks if the command is a background process
- Checks if the command has redirection
- Executes the command

```
...
if( background == 0){
    while (wait(&status) != pid);
    printf("Child process with id %d terminated\n", pid);
}
```

```
else{
    // add the pid to the background processes array
    int i = 0;
    while (processes[i] != NULL)
    {
        i++;
    }
    processes[i] = (process*)malloc(sizeof(process));
    processes[i]->pid = pid;
    processes[i]->name = newTokens[1];
    printf("Child process with id %d running in background\n", pid);
}
...
```

Contributing

This project was created as part of the Operating Systems course at IMC FH Krems. Therefore, we do not accept any contributions.

Authors

- David Bobek - 52107900 - 21imc11014@fh-krems.ac.at
- Dinu Scripnic - 52104494 - 21imc10070@fh-krems.ac.at
- Bertold Vinze - 52104496 - 21imc10072@fh-krems.ac.at